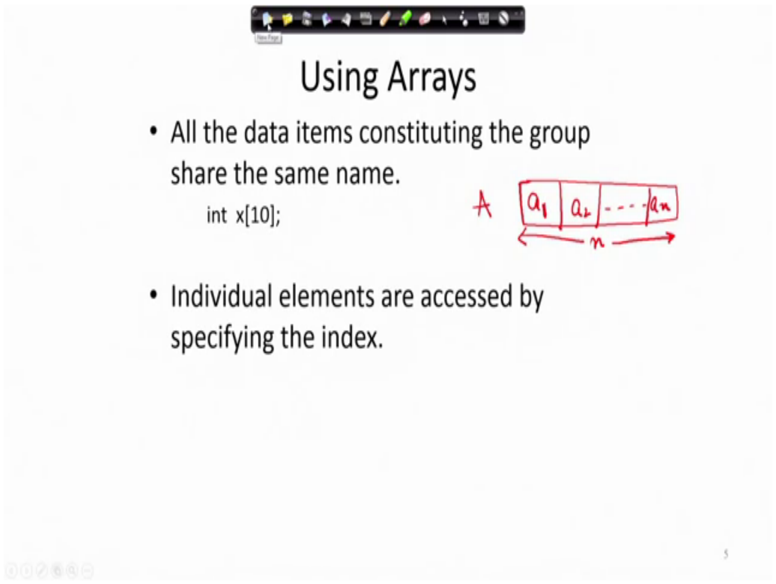
Problem Solving through Programming In C

Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 27
Arrays (Contd.)

So, we were looking at arrays and we have seen that all the data items constituting the array shared the same name.

(Refer Slide Time: 00:22)



Using Arrays

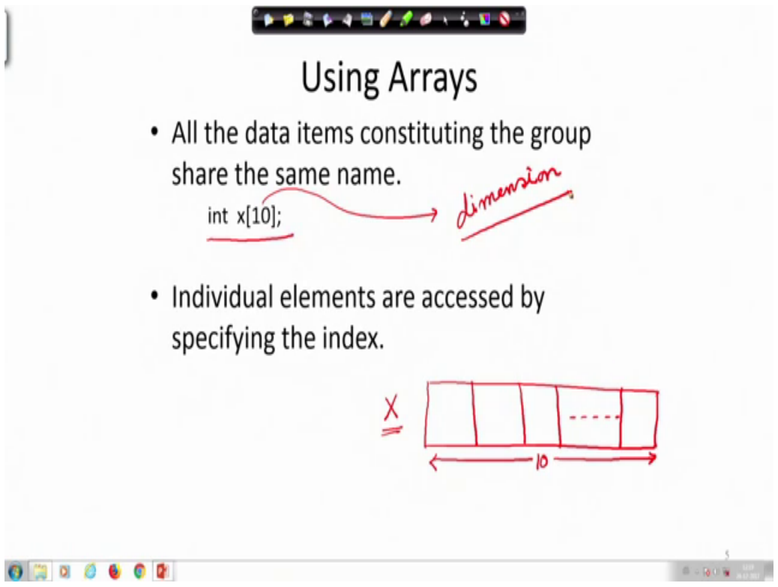
- All the data items constituting the group share the same name.

```
int x[10];
```

- Individual elements are accessed by specifying the index.

So, we can call an array to be A, an array A or a array num. If the array be A then the elements are a 1, a 2 like that up to a n, where n is the size of the array and each individual elements are being accessed by this indices, these are the different indices.

(Refer Slide Time: 01:11)



Using Arrays

- All the data items constituting the group share the same name.

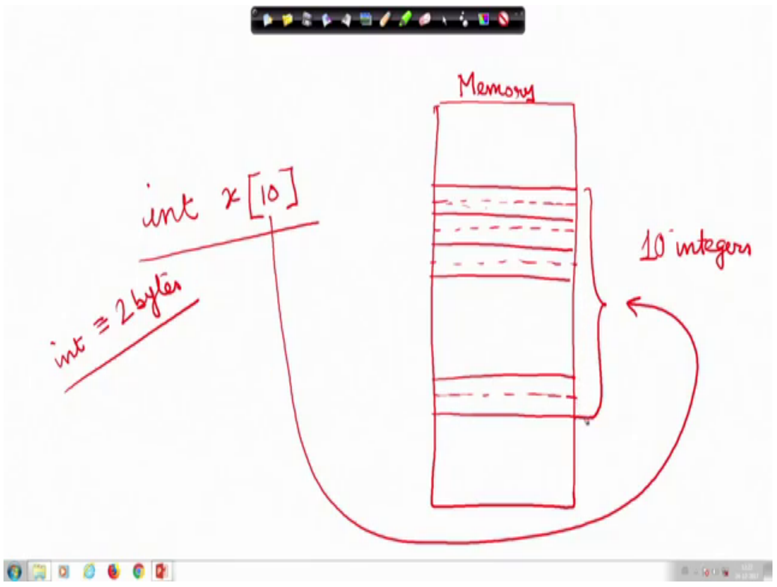
  int x[10];

- Individual elements are accessed by specifying the index.

So, for example, here when we declare here comes a new thing. Suppose we have an array where the size is 10, so there will be 10 elements of the array, 10 elements of the array. Just as and just like variables array as a whole is also a variable. So, suppose I name this array to be x these are x 1, x 2, x 3, x 4, x 5 like that up to x 10. Now, just like we had to declare the variables like int float here, here also we need to declare the array and the way we declare the array is shown here int x 10 what does it mean; that means, x is an array of now I am introducing a term of dimension 10. Earlier I was talking calling it size and here I am calling it dimension. For the time being assume that both these both of these are the same thing, but actually there is a difference, we will explain that in the course of discussion.

So, we declared it first of all the type of the array as I had said that an element all the elements of an array has to be of the same type. So, either all of them will be integers or all of them will be floats therefore, this entire array has is of a type int which says that all the elements of this array are integers and x 10. So, let us quickly have a look at the memory scenario. Say in the memory, this is my memory and this is the memory and in the memory some locations are kept for the array x; that means, start from here and maybe up to this.

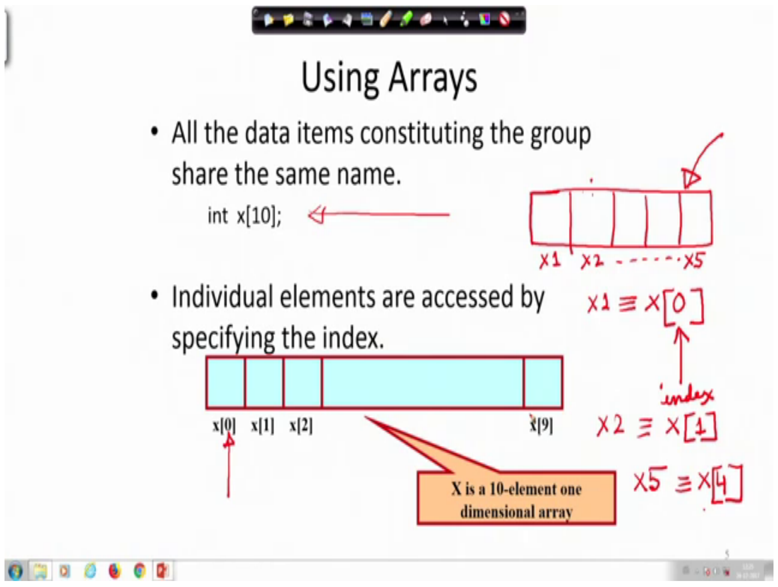
(Refer Slide Time: 03:31)



Now, how much space will be given to the array must be known to the compiler because the compiler just like the compiler allocates variables to different memory locations the compiler will also have to allocate locations memory locations for the array elements. Now, when I said int x 10; that means, the compiler will store space equivalent to 10 integers, 10 integers in the memory, 10 consecutive places for 10 consecutive one after another no gap elements, 10 consecutive integer places in the memory. Therefore, how many can you tell me, how many bytes are required for that? Integers and I have assumed that any every integer is taking two bytes, integer is taking two bytes. If I assume that then I will require 20 bytes here. So, there will be 20 locations each location consisting of two bytes in this way we will go on, right.

So, you must understand when I declare this, what is the meaning of this. Dimension of the array means that I am storing, I am reserving, I am reserving the compiler reserves the space for 10 integers and that has to be done before the array is used, before the array is used therefore, it must be declared before so that the compiler while it is compiling the program can allocate enough space.

(Refer Slide Time: 06:06)



Using Arrays

- All the data items constituting the group share the same name.

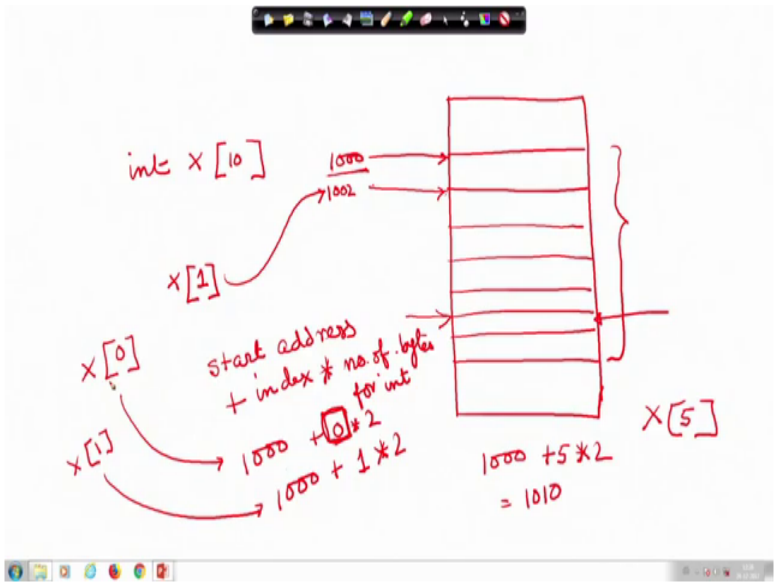
```
int x[10];
```

- Individual elements are accessed by specifying the index.

So, and now individual elements are accessed by specifying the index. So, here x, this x is a 10 element one dimensional array linear. Now, comes a peculiar thing if you look at this, we are saying that each of these elements are stored in these locations and which are identifiable and accessible by the indices. Now, in C language each index, the index value starts from 0. So, the first element although while discussing we are saying x 1 say I was drawing an array like this where I was saying that these elements are the first element x 1, x 2, that is true, x 3, x 4, x 5, but in C these elements are counted the counting is started from 0, there is a reason for that which will also be evident soon.

Just remember for the time being that the first element therefore, x 1 is actually represented in C as x then the square bracket followed by the index and the first one will be x 0, second one will be x 1, therefore, can you tell me what will be the last element of this array? How do I represent that? Yes, you are right x 5 will be nothing, but x 4 yeah. Since I am starting with 0 I can go up to 4. Here in this example as you can see that I have got 10 elements in the array therefore, I start from 0 and go up to 9 that is what is followed in C language. There is a reason for that let me just briefly mention the reason the reason would be this.

(Refer Slide Time: 08:54)

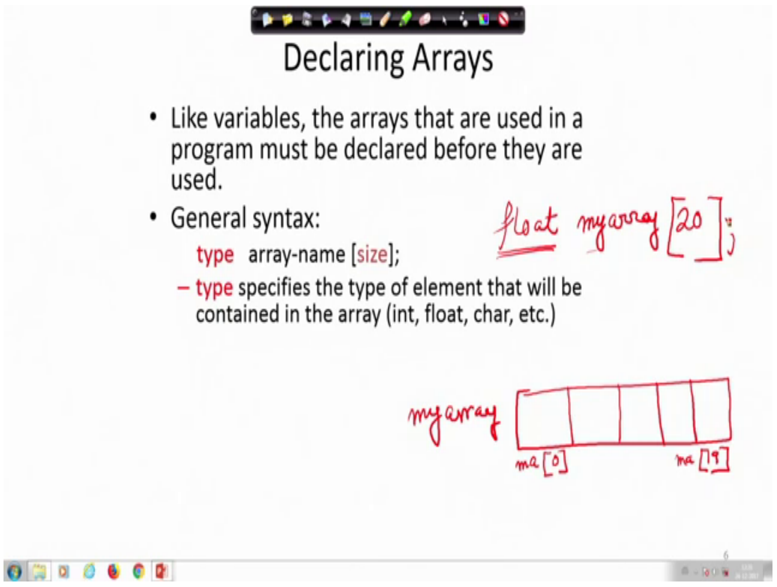


As I said that in the memory the array is stored in a particular region. When I declared it as int x 10 then 20 bytes have been given to me and the array starts from this point which has got some address maybe 1000 and this address is starting is 1002.

Now, and I know that each one is of size 2. So, when I say what is the address of x 1. What is x 1? x 1 means here and I know the starting of this array. So, start address plus index times number of bytes for integer will give me the actual address. So, for the first address what will be the index x 0? So, my start address is thousand for the first one thousand plus 0 times 2. So, that will be 1000 for x 1 1000 plus 1 times 2. So, that will be 1002, for x 5, x 5 in this way is which element; 6th element.

So, that one will be here 1 2 3 4 5 6, 1 2 3 4 5 6 this element. And what would be its address? The address will be 1000 plus 5 times 2; that means, 1010 it starts from 1010. So, 1002, 1004, 1006, 1008, 1010 here it will start. So, this is basically the offset. How much I shift from the top, that is the reason the ease of computation of the address of any of the index any index, any element with any element with a particular index can be done using this method this formula and therefore, we start with 0 that is a specific reason for starting with 0, all right. So, this is a 10 element one dimensional array that we have seen. Let us move ahead.

(Refer Slide Time: 12:22)



Declaring Arrays

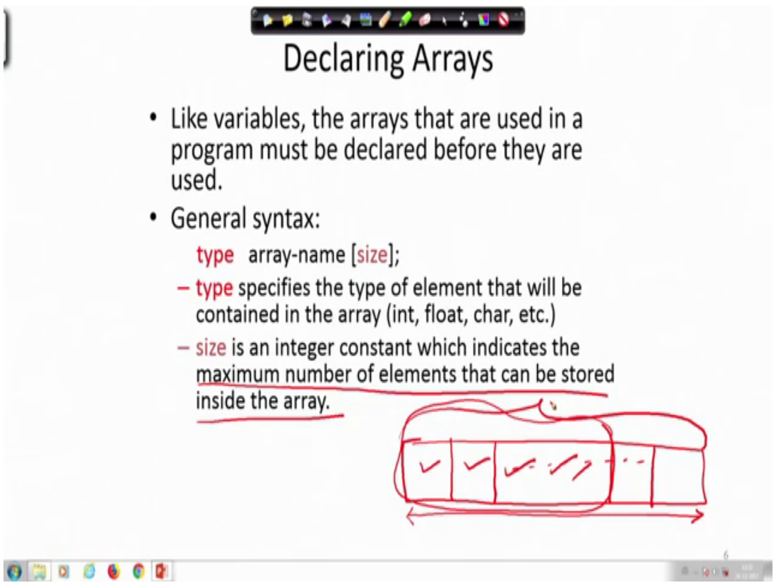
- Like variables, the arrays that are used in a program must be declared before they are used.
- General syntax:
  type array-name [size];
  - type specifies the type of element that will be contained in the array (int, float, char, etc.)

So, like variables the arrays that are used in a program must be declared before they are used the general syntax. Just like in integer or float we had declared them as in float before they are used. Why do I need to declare them before they are used? Because the compiler needs to allocate space for that otherwise the compiler does not know what type of variable it is therefore, we also have to do the same for the array and the general syntax will be type, array name, size. Now, again here the size I mean the dimension. Type specifies the type of the element that will be contained in the array it can be int, float, char or whatever.

So, I have got fields like say here I can have an array I can dictate as float the array name can be my array and size may be 20. Now, this one specifies that this array can hold only floating point numbers, and suppose floating point numbers are 4 bytes each then for how much memory will be reserved for this 80 bytes will be reserved for this because 20 is a size; that means, the maximum size that the array can take. And what is my array? Everything a just like every variable must have a name this array my this area also has got a name called my array. And so what will be the indices? I am repeating the thing my array is 0. So, I am writing ma for short 0 and what would be the last element here ma for size 20, it will be 19, is it all right.

So, this is the general syntax which a must put and also there should be a semicolon at the end, just like other declaration there is no other major difference here.

(Refer Slide Time: 15:04)



Declaring Arrays

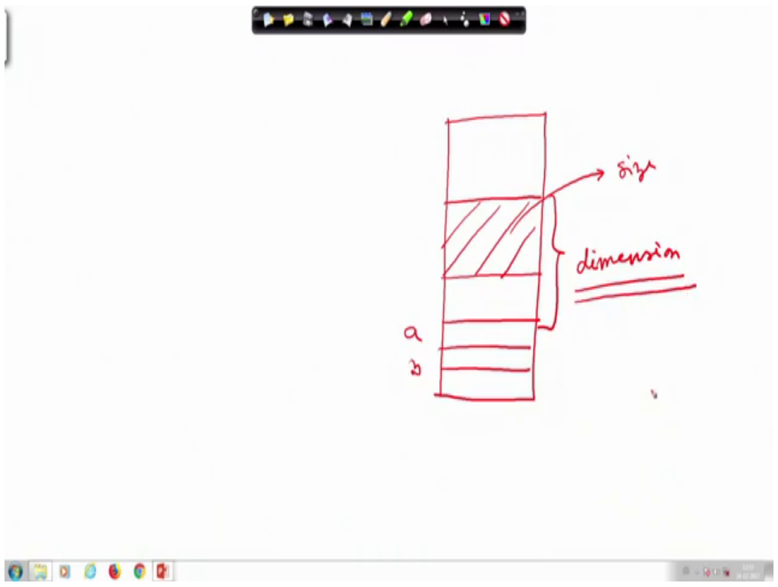
- Like variables, the arrays that are used in a program must be declared before they are used.
- General syntax:

  type array-name [size];

  - type specifies the type of element that will be contained in the array (int, float, char, etc.)
  - size is an integer constant which indicates the maximum number of elements that can be stored inside the array.

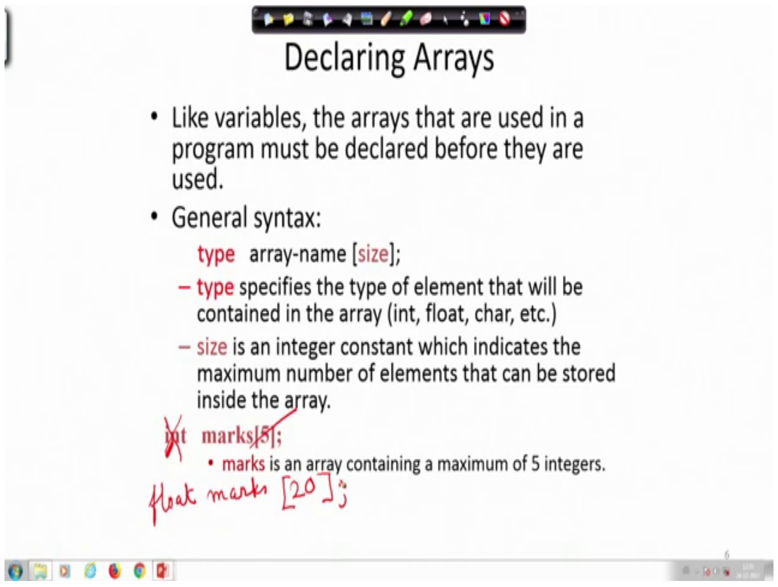
Size is an integer constant here I am calling it dimension that is it is a maximum number of elements that can be stored inside the array. Suppose I have got so much space kept for storing an array all right, so much space many many elements can be stored I have reserved so much space, but in my actual working I am using only some spaces and these are not touched that is allowed that means, but the reverse is not allowed. Unless I can, I have got the space unless I have got space I cannot store them. So, I must reserve the space beforehand but. So, that is why I want to I was mentioning this maximum space as dimension and the actual number of spaces which you are using to be size of the array all right.

(Refer Slide Time: 16:21)



The reverse is not true because suppose I have an; I have reserved. So, much space for an array and I go I go beyond that then these marks are spaces for other variables like a b c, their data will be destroyed by my data which I am taking as an array. Therefore, we must restrict to this, but suppose out of this I am using only this much in my parlance I am calling it size and this to be the dimension you can call it in this example in the slide we are calling this to be the size of the array there is a maximum size of them. So, what is dimension? Dimension is the maximum size that the array can be off, but in practice in actual running it can be less than that, but not more than that.

(Refer Slide Time: 17:22)



Declaring Arrays

- Like variables, the arrays that are used in a program must be declared before they are used.
- General syntax:

  type array-name [size];

  - type specifies the type of element that will be contained in the array (int, float, char, etc.)
  - size is an integer constant which indicates the maximum number of elements that can be stored inside the array.

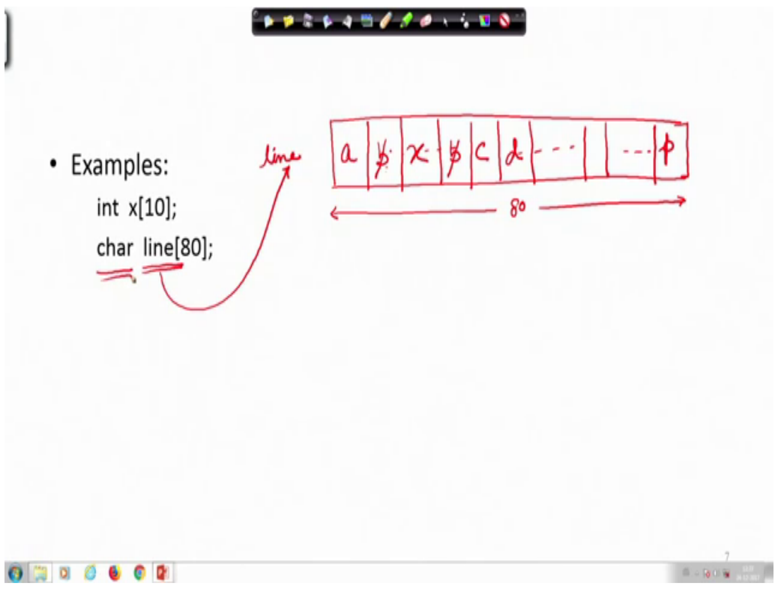
~~int~~ marks[~~5~~];
  - marks is an array containing a maximum of 5 integers.

float marks [20];

So, I hope that part is clear to you. So, here what did I write? int marks 5 what does it mean? It means I am storing marks and the marks are all integers right 50, 55, 60, 100 I am storing integer marks; and how many marks can I store? 5. So, what will be the indices? 0 marks 0 to marks 4 that is the index limit because I am starting from 0 I can have only 5 positions. So, marks is an array containing maximum of 5 integers.

If a teacher decides that he will give fractional marks that somebody can get 62.5 somebody can get 70.2, somebody can get 59.7 then what change should we do the change that we should do is this will be replaced with float. And if I say in my class there can be maximum 20 students then what else should I change, I should change this and make it 20 that is how I should declare all right.

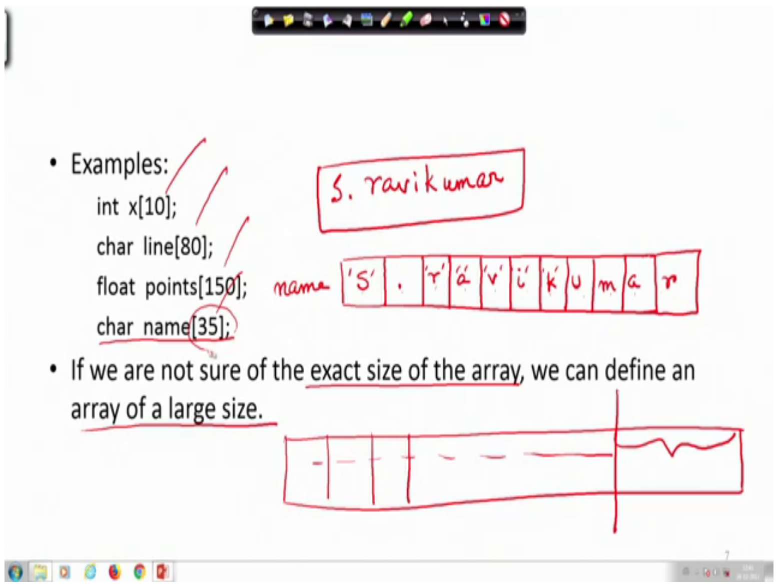
(Refer Slide Time: 18:58)


- Examples:
  - int x[10];
  - char line[80];

Here are few examples int x 10; that means, x is an integer. How do we read it? x is an integer x is an array of integers of size or dimension 10. char line 80; that means what? that means, line is a variable array of type character. So, what will it be? There will be 80 such positions usually when we take a printout usually when we take a printout of characters the lines were conventionally 80 characters in a line. So, there will be there will be positions like this all through maximum 80 all right. So, the size will be 80 and this variable is known as a line and what can it hold it can hold a characters like a, sorry a there can be a space, space is also a character I denote space as a blank like this then maybe x is a character, then there is a blank a blank is also varied character then c, then d, all those things can be there.

So, it is an array of say ultimately say p is the last character something like this. Altogether 80 spaces for character I keep in my variable line and what is the type of the variable that type of the variable is an array, array of character all right.

(Refer Slide Time: 21:03)



- Examples:
  - int x[10];
  - char line[80];
  - float points[150];
  - char name[35];
- If we are not sure of the exact size of the array, we can define an array of a large size.

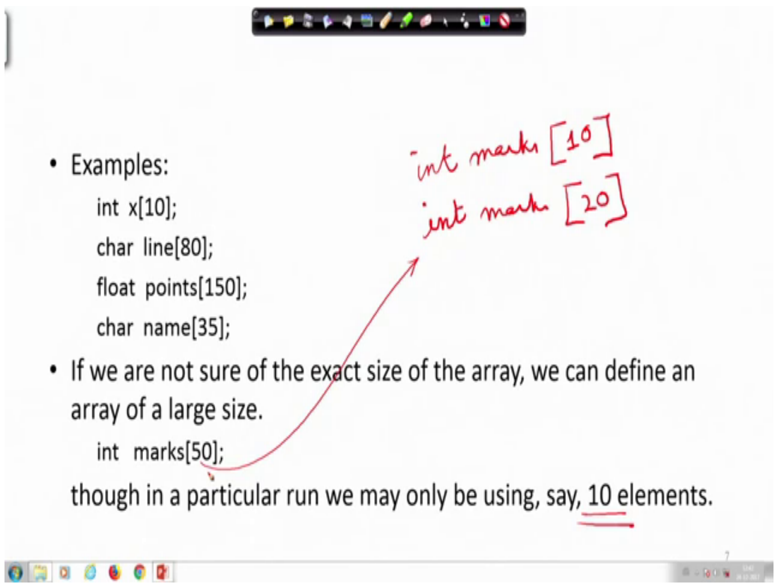
Similarly, I am saying that there are 150 points, there can be 150 points and each of those points are floating point numbers. Say name, I just stored a name, I want to store the name of a person say name S Ravikumar is the name of a student and I want to store it in a computer. How do I store it? I can store it in the form of an array where each of the elements is a character first one is S then dot then r then a v i k u m a and I need one more space r.

So, you see I needed one more space here. Why? Because I had taken I did not have enough space here. Now, each of them is a character by the way you know that a character is denoted like this so and so forth. Now, what this line says is that name is a variable which has got that the capability of holding 35 such characters, it has got the ability of storing 35 such characters at the most.

Now, how much space should we keep? How much should I keep here? If we are not sure of the exact size of the array we can define an area of large size. Say if I think if I have got no idea of how the what are the typical Indian names for example, or American names for example, I can keep name 100, but is it advisable not always. When I really do not have any idea I have got no other way, but to do it, but if I have an idea of how much the name length can be at the most I should keep so much size because if I keep an arbitrarily large size then what am I wasting, I am wasting my memory space because each of these are a memory location.

Suppose beyond this never a name can extent to then I should not keep this part in the name. But when I have got no idea of course, I have to keep a bigger large size. So, that is the difference between the actual dimension of the array how much reserve we do and for a S Kumar, S Ravikumar for example, my actual size is 1 2 3 4 5 6 7 8 9 10 11 all right out of 35, 35 was the available number of spaces.

(Refer Slide Time: 24:43)

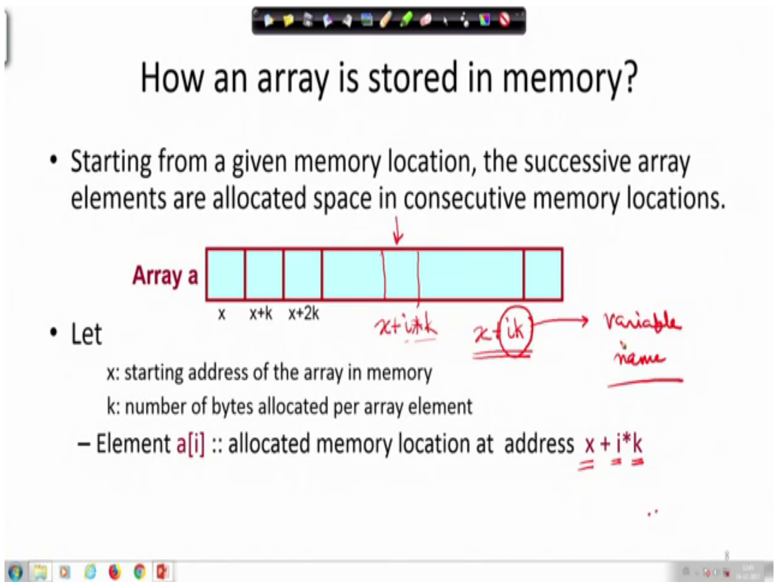


- Examples:
  - int x[10];
  - char line[80];
  - float points[150];
  - char name[35];
- If we are not sure of the exact size of the array, we can define an array of a large size.
  - int marks[50];

  though in a particular run we may only be using, say, 10 elements.

So, when we say it is int marks 50. Suppose in my class I have got say 10 students and I am going to store the marks of only one subject in that case there is no point storing so much space I mean it is like reserving so much space for the variable marks. If I know since I know only 10 students are there, if I know beforehand then I can I could have written int marks 10 or if I had known that sometimes the number of students in the class are 10 sometimes 15, but never more than 20 then I could have kept marks 20, but I should not keep marks 50. I hope the point is clear now.

(Refer Slide Time: 25:40)



How an array is stored in memory?

- Starting from a given memory location, the successive array elements are allocated space in consecutive memory locations.

- Let

  x: starting address of the array in memory

  k: number of bytes allocated per array element

  – Element a[i] :: allocated memory location at address x + i*k

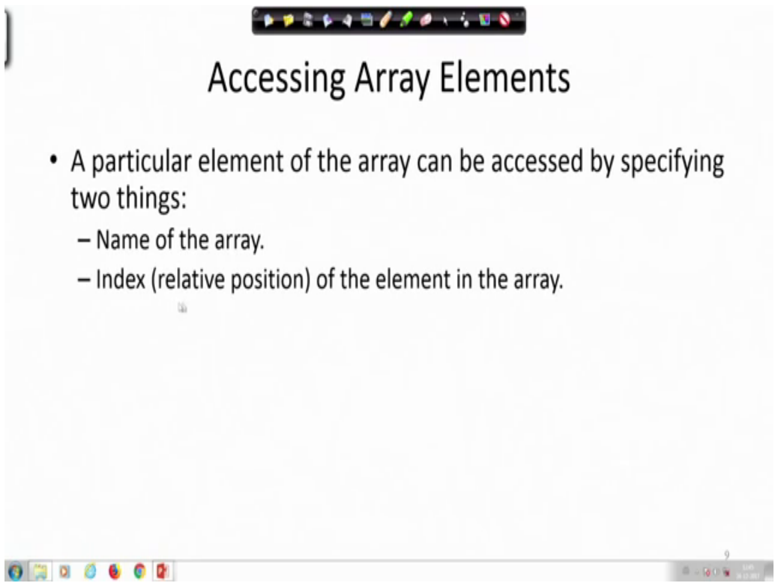
Now, I have already described this how an array is stored in a memory. Starting from a given memory just let us have a revision of this, starting from a given memory location the successive array locations this is very important, the successive array elements are allocated spaces in consecutive memory locations one after another.

So, array a will have memory locations one after another like this let now the same computation let us do it again. Let x be the starting address of the array sorry let x be the starting address of the area address, this is the address and k is the number of bytes allocated power array element I had shown it for an integer now I am showing it in a generalized form, x was the first starting location say 1000 or whatever and k is the number of bytes allocated.

Now, then the next one will be x plus k because it started with x and needed k locations the next one will start from x plus k, the next one will start from x plus 2 k like that the ith element therefore, will be as we had computed earlier will be x plus i k and since we started with 0 that will be perfectly all right. Element a i will be located some a i will be located the address say a i here will be located starting at x plus i times k. Just a little bit of puzzle here I have written it here. Like this a common mistake could be that I what would have happened if I had written it like this. What is the problem of writing in this way? the problem is; what is i k? i k is not i times k not i multiplied by k, but i k is another variable name all right.

So, these two are not the same thing. Whenever we go for programming we should be very careful. So, first array index is assumed to start at 0, in C we are doing that.

(Refer Slide Time: 28:24)



Accessing Array Elements

- A particular element of the array can be accessed by specifying two things:
  - Name of the array.
  - Index (relative position) of the element in the array.

So, a particularly element of an array can be accessed by specifying two things. What are the two things? Name of the array? Index, index is nothing but the relative position of the array. We will continue with this discussion further.